METHODICAL DESIGN OF COMPUTER-AIDED MACHINE DESIGN SYSTEMS

G. KARSAI

Institute of Machine Design,
Technical University, H-1521 Budapest

Received June 20, 1984
Presented by Prof. Dr. L. VARGA

Summary

A design principle for preparing mechanical construction design systems has been suggested, starting from a possible algorithm of conventional design, and leading to, function structure and programming language of the computer-aided design system implementing the algorithm. A new, kinematic approach to geometry design has been applied.

Introduction

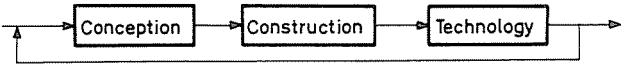
Since the beginnings of designing activities until quite recently, several attempts have been made to comprehensively describe the process of designing—within this, machine designing—and to elaborate its methodology, i.e. "to design designing". This problem has been particularly emphasized since the advent of computers in engineering. It is of course somehow solved by every designing system. The majority of systems with extended uses undergo several transformations during their lives and generally grow ever bigger and ever more complex. However, this complexity soon turns into an obstacle to development and utilization. The crisis can only be surmounted by reconsidering the fundamentals from time to time.

In this study a design principle is proposed for preparing mechanical construction design systems, making use of achievements of methodical machine designing [1], structured programming [2], and of systematics in general. In the first part, a possible algorithm of machine designing, and the function structure of the implementing design system are outlined, proceeding from abstract to concrete, from general to individual. The second part deals with the formalization of the outlined algorithm and implementation in a programming language. A new, kinematic approach is used for the formalization of geometry design, decisive for mechanical systems [3, 4].

Substantial representation of construction design

Cycle of designing

At the present stage of labour division, at least three important design stages can be distinguished



The cycle of designing is an open system, part system of the cycle of reproduction, or, as commonly called, of the cycle of innovation. Even its subsystems are complicated systems, with feedback, and even prediction.

a) The phase of *conception* is that of comprehensive ideas, economical and technical considerations, from overall alternatives to decision-making.

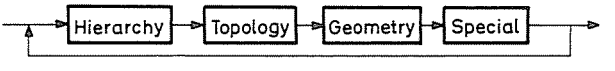
b) The *construction* design, by conventional name product design, means realization and detailed design of the system taken shape in the previous phase.

c) The design of *technology* or production design means to combine facilities and methods, kinds of materials and energy, in order to (optimally) materialize the design of construction.

In the order from a) to c), the ratio of heuristic to manual gives decreasing. Consequently, the conceptual phase is the least, and the technological phase the most accessible to computerization. On the following pages, however, only design in the narrow sense, design of construction, will be discussed in detail.

Cycle of design of construction

Subsystems of design of construction are advisably discerned according to the functional structure of the final result of designing, the complete design documentation (lists, drawings of assembly and components, technical calculations, etc).



a) The subsystem called *hierarchy* involves an administrative activity of classification, and makes it possible to describe the construction as a set (e.g. of elements forming a subunit, subunits forming a structural unit, and so on). In conformity with traditional concepts, this phase corresponds to the description on (piece) list level.

b) The *topological* subsystem represents qualitative relationships between the constructional elements, showing how and along which surfaces of action (subspaces) the parts unite to form a whole (e.g. how to assemble a subunit from its components). This phase corresponds to the level of schematic diagrams, fitting and *assembly drawings*, or on a lower level of elaboration, to flowcharts, kinematic sketches, design sheets, etc.

c) The subsystem of *geometry* describes the construction as a rigid body, essentially on the level of component drawing. *It is determinant in mechanical systems*, because the majority of information arises in this phase.

d) The subsystem called *special* represents *technical calculations* to be included in the technical documentation of the construction.

The description corresponding to phase a) is the roughest. Though, knowledge of the "family tree" of a product permits to mechanize several other activities (stockpiling, preparing orders, etc.).

Systems that include only standard, typified building units can be almost entirely described in phases a) and b). Although it is a natural endeavour to develop building-case systems, pure modular systems are rare in mechanical engineering, so that designing cannot generally be limited to these two phases.

Nowadays the most current and refined form of communicating the information included in phases a), b), and c) is machine drawing. However, computer-aided design requires other, coded forms. These representations are no longer merely alphanumeric or graphic, but they constitute abstract mathematical information as *direct, initial data for the subsequent automated phases of design and production* rather than passive, visual display of the documentation. (It is a mistake to believe that the chief attraction of the computerization of designing is the possibility of automatical documentation. The system of traditional paper documentation will probably cease to exist in the long run.)

The majority of designing methods (phase d) are actually specific to the particular speciality. Using computers, however, these models can be made more complex, more complete, i.e., more exact and reliable than those accessible to manual handling.

The reader is most likely to recognize traditional design phases. This is obviously not accidental. The computer-aided design basically differs from the traditional one only in its means: its function and essence (i.e., that it is a conceptual realization of a material system) are the same. The interaction between means and method is obviously unquestionable, but this is a question of secondary importance from our point of view.

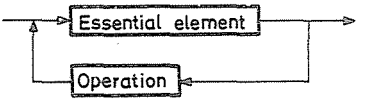
The decisive majority of processes in engineering systems are fixed, forced changes arising in systems to be considered as rigid bodies in first approximation. *Thus, most of the technical information are of mechanical type and within this, of geometrical character.* For that very reason, the geometry subsystem will be set forth in detail.

Design of geometry

The design of geometry has several definitions that are equivalent as concerns the final result. One of the simplest is the following:

Essential element

Field element



Thus, the following activities are included in *the cycle of geometry design.*

a) Choosing an essential element of appropriate form dimension and position, and

b) generating compound field elements as a complex of essential elements connected by set operations.

The most frequent essential elements are half-spaces limited by plane, cylindrical, conical and spherical surfaces, and the operations are generally the formation of unions, intersections and complements. Higher developed solutions permit the hierarchic generation of field elements, and their treatment as essential elements.

The range of essential elements is specific to the particular speciality; sometimes (e.g. for piping designs) they are graphic symbols rather than geometrical objects. Since in machine drawing, the set operations are not explicited either, further on only *the determination of dimensions and positions* will be dealt with. The outlined approach does not sharply distinguish even these two.

The determination of positions (and dimensions) mathematically implies the specification of a special affine transformation. Namely, relative position and orientation of any two rigid bodies can be described by a secondary tensor (hypermatrix),

$$\mathbb{T}=\left[\begin{array}{c|c} 1 & 0^* \\ \hline d & \mathbb{D} \end{array}\right]$$

where d is a vector connecting a characteristic point of each of the bodies (translation), $\mathbb{D}$ is an orthogonal matrix (rotation), maybe their scalar multiplication (scaling).

As all real dimensional nets are coherent, any two essential elements (m and n) can be connected by a *dimensional link*. The resultant of the dimensional link is unambiguously determined by product

$$\mathbf{T}_{mn} = \mathbf{T}_{mi}\mathbf{T}_{ij} \ldots \mathbf{T}_{kl}\mathbf{T}_{ln}$$

since the dimensional nets permitted by the machine drawing are loopless (tree structures, with transformations T assigned to the edges).

In fact however, even in the simplest cases the concept of unknown (e.g. resultant) dimensions has to be introduced and calculated (e.g. typical dimension of certain elements is a resultant dimension, or the technological dimensions have to be determined from constructional ones). Completing the given dimensions with the unknown ones results in loops in the dimensional net. Along the loops (closed paths) the resultant of the dimensional link is inevitably the unit transformation:

$$\mathbf{T}_{mi}\mathbf{T}_{ij} \ldots \mathbf{T}_{kl}\mathbf{T}_{lm} = \Pi\mathbf{T}_{ij} = \mathbf{T}$$

The "*geometry loop law*" provides 3, and 6, independent equations in plane, and in space, resp., loop-wise, yielding as many unknown linear and angular dimensions (unless the system of equations is a degenerate one). In the general case the solution is multivalued.

The dimensional net can also be regarded as a geometrically (kinematically) determined system, with zero degree of freedom. The systems with one, two and three degrees of freedom describe lines, surfaces and subspaces, respectively. Choosing these subspaces as essential elements of geometry the geometry design is easy to formalize and automate with methods of systems theory (net theory). For the calculation excess compared to the analytic, polynomial spline and other representations, practically predominant nowadays, an ample compensation is to bring computer-aided design nearer to traditional engineering approach also true for technology design. The point is not only to generate combinations of linear and circular movements in the simplest and most precise way, but above all, to *describe the kinematics of machine-tools, robots, manufacturing cells, etc., by the same formalism as the geometry and kinematics of the product.*

Many of the physical and other theories can be formulated on the basis of systems theory (metatheory), at most the given phenomenon is of continuous character, or the tensorial order of magnitudes is different. Such a structure of subsystems has some concrete, practical advantages in addition to the homogeneity of the formalism.

In the cycle of designing, the description of the construction grows ever richer and sophisticated. Consequently, the design systems have to be universal, "obvious" and adaptable enough to follow *the dynamism of the design process.* In the following, the programming language of such a system of mechanical construction is outlined, gradually developed and enriched to perform the global functions above. The language can be supplemented for special functions; the stress is laid on the logical structure of the language, or rather on the things it maps.

Formal representation of construction design

Every formal system is an abstract model making the intuitive concepts accessible to mathematical means. A formal system is, for example, a programming language, but such systems are also used for defining these languages. The same will be done in the following. The metalanguage underlying the grammar of designing is a syntax-graph variant of the *Beckus-Naur form.* Proceeding along the syntax graph, results in syntactically correct programs.

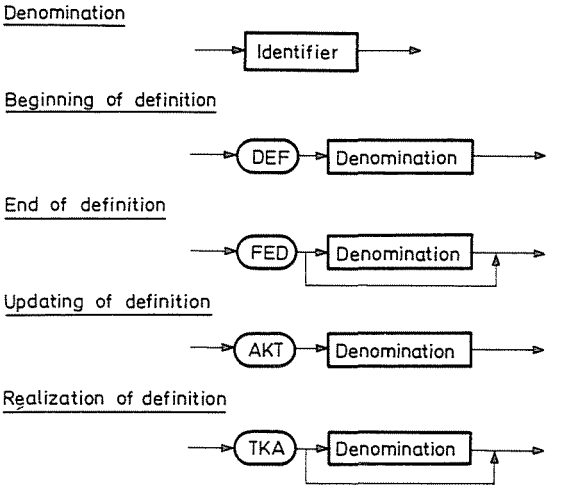
In the following the syntax (formal rules) and semantics (contents, meaning, interpretations) of the "constructional language" is built up to nontrivial complexity according to the above principles.

Hierarchy of structure

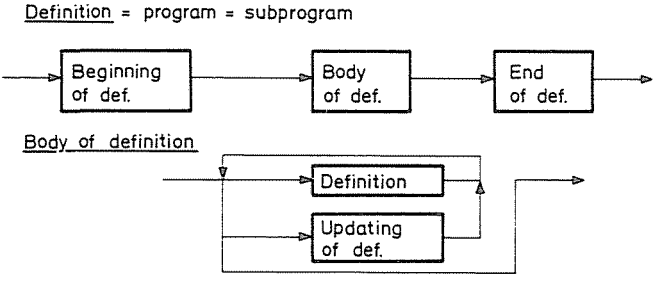
A description on list level is basically a relation interpreted on the set of components (essential elements, primitives). Obviously this structure is inevitably a *hierarchical, so-called tree structure. Ready-made models* for such systems are programming languages permitting to develop subprograms (begin-end, subroutine, procedure, macro, etc.). Such *structured programming languages are* e.g., Algol, Pascal, C, and the most of high-level programming languages.

To describe hierarchy, four linguistic invariables (key words, terminal symbols) and a variable (grammatical category, nonterminal symbol) will be introduced. The beginning and the end of each definition (subprogram) are denoted by key words DEF, and FED, respectively. Symbol AKT is used for calling or setting forth the definitions, symbol TKA for performing (running) the program (subprogram). The only grammatical category is the denomination, called identifier, of the structural units.

The instructions are formed according to the following rule.



Any well-known rule of forming identifiers can be chosen. The program is not distinguished from the definition, to grant the possibility of subsequent use as a subprogram. Let us notice that this is a recursive syntax. On the other

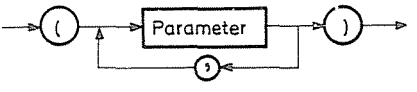


hand, *the program* itself *cannot include definitions updating themselves, or (recursive) definitions mutually updating one another.* Namely, a finite subsystem cannot be a subsystem of its own (Russel's paradox).

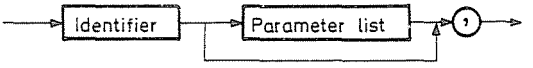
The definitions obviously cannot overlap, and only definitions of the same or higher level can be updated.

Topology of structure

Strictly speaking, the topology of structure is a relation qualitatively characterizing concrete physical and other relationships between subsystems. In order to describe it, three more terminal symbols, parentheses "( )", separating character ",", as well as a nonterminal symbol, the name (identifier) of the connections are introduced. These symbols make up a parameter list:
Parameter = name of connection = identifier
Parameter list



Now, the concept of denomination is expanded as follows.
Denomination



In the instruction "def." the parameters are formal, alibi names, they are concretely defined in the series of the instructions "akt.". *The parts unite into a whole along their namesake connections.* All identifiers but parameters are local. It is obvious that in this way structures with any kind of *loops* can be coded.

The essential element (primitive) of hierarchy is a definition exempt from references to other definitions (these are "the leaves of the family tree"). However, from the aspects of topology and geometry, and so on, also essential elements are complex things with an internal structure of their own. Let us see now an example of how to describe the geometrical structure of elements, restricted to dimensional nets.

Geometry of structure

Let TX, TY, TZ stand for *a unit length displacements* (translations) along a coordinate axis defined in the ordinary Euclidean space, and RX, RY, RZ for *unit angle turns* (rotations). Furthermore, let S be a positive integer. It is easy to realize that an S-fold displacement or rotation equals the S^{th} power of unit transformation. For instance:

$$TXS = TX * TX * \ldots * TX = TX \uparrow S,$$
$$RXS = RX * RX * \ldots * RX = RX \uparrow S.$$

It is clear that the relationship can be generalized to real S exponents, so that a dimensional link in mathematical form looks, like e.g.:

$$TX\uparrow 9.5*RY\uparrow -45*(TX\uparrow S*RZ\uparrow 0.5*TX\uparrow S)\uparrow 60$$

Exponent S is *a numerical variable*, identifier of a symbolic dimension. Parentheses function as usual.

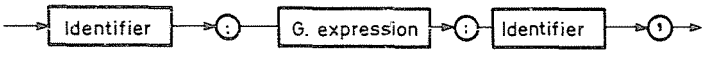
Since there can be branches in the dimensional net, the concepts of start and end points of the dimensional link have to be introduced:

$$P1:TX\uparrow 9.5*\ldots\uparrow 60:P2,$$

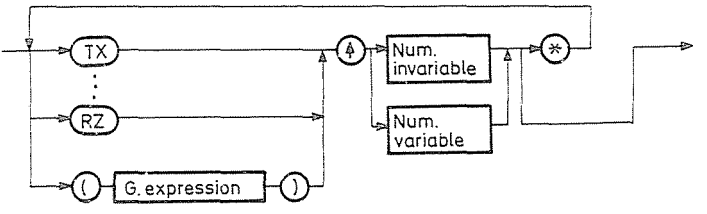
where P1 and P2 are *the identifiers* (*labels*) *of the start and end points*, i.e., the names of connections. Contrary to the usual serial programs, several instructions may have identical labels.

Relying upon the examples, it is no longer difficult to create the syntax of geometrical instructions:

Geometrical instruction



Geometrical expression



Compound geometrical expressions have recursive definitions. It is advisable to let geometrical instructions occur in the body of any definition (this requires a minor modification of the rules). The syntax of the numerical variable is identical with that of the identifier, and the syntax of the numerical invariable can be chosen conventionally. Numerical variables can also be included in the list of formal parameters, and invariable dimensions can also occur among the actual parameters. So the resultant (or unknown) dimensions are the numerical variables which do not get any value when updated. One problem of the geometry system is to calculate these dimensions.

The outlined syntax is *not the only* possibility, it is only an alternative. For example, operational characters ($*$ and $\uparrow$) can be omitted from the geometrical

instruction, and parentheses can be substituted for ":". In three dimensions it is enough to retain one displacement and two rotational operators, in two dimensions a single rotational operator is enough, besides displacement. A scale operator, or scaling along all the three axes can be introduced, (although the latter tends out of the domain of Euclidean transformations). It is obvious that the set of instructions can be changed and completed as needed.

The language has not been prepared for self-contained man-machine dialogue alone, it can be used as e.g. an intermediate code for alphagraphic and other interfaces, or embedded in certain high-level programming languages.

The given formalism prefers the principle of analytic design that starts from the whole and reduces it to its component parts. At the same time it makes it possible to follow a reverse order of realizing the design (coding, modification, testing, etc.), i.e. to synthesize. The feedback realized in the cycle of design actually permits *any combination of the analytic and synthetic types of activities.*

Computers can directly perform only tasks formulated on language basis, in the form of code combinations similar to those above. Though, designers consider e.g. computer plotting a language that is not a computer language in the strict sense of the word, although for the designers this is the most natural form of communication. The designers of designing systems ask the design engineers for indulgence. For the time being, there are no devices for completely and automatically solving most of the problems of designing.

References

1. PAHL, B.—BEITZ, W.: Konstruktionslehre Springer-Verlag, 1977.
2. DAHL, O. J.—DIJKSTRA, E. W.—HOARE, C. A. N.: Structured programming. Academic Press, 1972.
3. DENAVIT, J.—HARTENBERG, R. S.: Journal of Applied Mechanics, June, 215 (1955).
4. KARSAI, G.: Koncepcióvázlat gépészeti tervező-rendszerek készítéséhez. Kézirat, Budapesti Műszaki Egyetem, Gépszerkezettani Intézet, 1983.

Dr. Géza KARSAI H-1521 Budapest